
DIGITAL ACCESS TO SCHOLARSHIP AT HARVARD
DASH.HARVARD.EDU


HARVARD LIBRARY
Office for Scholarly Communication

A writer's collaborative assistant

Citation
Tamara Babaian, Barbara J. Grosz, and Stuart M. Shieber. A writer's collaborative assistant. In Proceedings of the Intelligent User Interfaces Conference, pages 7-14, San Francisco, CA, January 2002. ACM Press.

Published Version
http://doi.acm.org/10.1145/502716.502722

Permanent link
http://nrs.harvard.edu/urn-3:HUL.InstRepos:2252600

Terms of Use
This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA

Share Your Story

The Harvard community has made this article openly available.
Please share how this access benefits you. Submit a story .

Accessibility

A Writer's Collaborative Assistant

Tamara Babaian
CIS Dept., Bentley College
Waltham, MA 02452
tbabaian@bentley.edu

Barbara J. Grosz
DEAS, Harvard University
Cambridge, MA 02138
grosz@deas.harvard.edu

Stuart M. Shieber
DEAS, Harvard University
Cambridge, MA 02138
shieber@deas.harvard.edu

Abstract

In traditional human-computer interfaces, a human master directs a computer system as a servant, telling it not only what to do, but also how to do it. Collaborative interfaces attempt to realign the roles, making the participants collaborators in solving the person's problem. This paper describes Writer's Aid, a system that deploys AI planning techniques to enable it to serve as an author's collaborative assistant. Writer's Aid differs from previous collaborative interfaces in both the kinds of actions the system partner takes and the underlying technology it uses to do so. While an author writes a document, Writer's Aid helps in identifying and inserting citation keys and by autonomously finding and caching potentially relevant papers and their associated bibliographic information from various on-line sources. This autonomy, enabled by the use of a planning system at the core of Writer's Aid, distinguishes this system from other collaborative interfaces. The collaborative design and its division of labor result in more efficient operation: faster and easier writing on the user's part and more effective information gathering on the part of the system. Subjects in our laboratory user study found the system effective and the interface intuitive and easy to use.

1. Introduction and Motivation

In traditional human-computer interfaces, a person acts as the master directing a computer-system servant. Collaborative interfaces [17] attempt to realign the roles, making the participants collaborators in solving the user's problem. Formal models of collaboration [5, 8, 7] identify as some of the key features of a collaborative activity commitment to a shared, or joint, goal; an agreed-on division of labor; and communication between the parties to enable the satisfaction of joint goals. Whereas in a traditional interface the human user is the repository of all goals and takes all the initiative in determining ways to satisfy them, in a collaborative interface the participants establish shared goals and both take initiative in satisfying them.

For example, the GLIDE system [16] is a network-diagram layout tool in which the user and the computer simultaneously and seamlessly work to satisfy the user's layout goals. Goal-sharing is achieved by the user's conveying layout goals through direct manipulation, and the division of labor in achieving the goals is implicit in the design of the system as a whole. Thus, a level of collaboration is achieved without explicit reasoning about goals or the state of the world. The Distributed Information Access for Learning (DIAL) system [13] provides for multi-media interactions with a complex information system; DIAL works with users to identify information relevant to their needs. The manner in which DIAL interacts collaboratively derives from the SharedPlans theory of collaboration [7]. DIAL uses explicit representations of recipes for domain actions and reasons about intentional contexts to lessen the amount of information a user needs to provide in querying the system. It demonstrates both the efficacy of deploying a model of collaboration to inform the design of a system and the system limitations that arise from limited reasoning about knowledge and actions.

GLIDE and DIAL were designed to directly implement key features of a formal model of collaboration, handling various belief and intentional constructs implicitly. The formal model of collaboration is used as a *design guide* in the design of the system, but is not reasoned with directly. An alternative design philosophy is found in the Collagen system [14], in which the formal model is directly reasoned with, mechanisms are incorporated to manage databases of beliefs and intentions, and a recipe library of predefined plans is used. In this case, the formal model of collaboration is treated as a *specification* of the implementation.

In this paper, we explore another part of the design space of collaborative interfaces. We describe a writer's collaborative assistant, implemented in a system called Writer's Aid, designed to support an author's writing efforts by performing various bibliographic tasks that typically arise in the process of writing a research manuscript. As in GLIDE and DIAL, Writer's Aid follows the design-guide approach. Also like earlier systems, the division of labor between the user and Writer's Aid is predefined and constant. A distinguishing feature of Writer's Aid is its ability to autonomously generate and execute plans to achieve goals provided by the user and adopted by the system. This autonomy, enabled by use of automated planning, also distinguishes Writer's Aid from other collaborative interfaces with predefined recipes. It en-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
IUI '02, January 13-16, 2002, San Francisco, California, USA
Copyright 2002 ACM 1-58113-459-2/02/0001 ...$5.00.

ables Writer's Aid to act as a robust collaborative partner, undertaking tasks in the service of a joint goal (producing a manuscript with well-formed citations) and pursuing all known avenues to accomplish those tasks.

The use of planning to organize the behavior of a collaborative system is especially important in tasks for which there is more than one possible course of action and where some of the actions may unpredictably fail. Dealing with bibliographic records and papers is one such problem domain. Papers and bibliographic information are often available from multiple electronic sources such as digital libraries, author's homepages, and on-line bibliographies. It is burdensome for a person to search systematically and thoroughly different sources to locate papers and tedious for people to compose bibliographic records. Because Internet searches are typically incomplete, many authors also must consult hard copies of journals and conference proceedings. The creation of citations is also disruptive to the writing process. Most of such work is more appropriately done by a computer system that can plan for a wide variety of approaches to data gathering and pursue them exhaustively. Similarly, many actions, such as accessing bibliogaphic databases or web resources, can fail (for instance, due to a server failure). In such a case, a planner can dynamically recover and replan, efficiently reusing already obtained information, until a goal is satisfied or all ways of satisfying it fail.

Planning has proven advantages in the task of information integration from multiple distributed sources; it hides from the user the process of data acquisition and manipulation [1, 10]. We take this idea further and weave such information integration into an ongoing human-computer collaboration on a broader task that is the source of the information need. This setup creates advantages for both parties and thus results in more efficient overall execution of the task. The user's simultaneous involvement in editing the paper and expertise in the particular academic field provides the computer-assistant with highly selective query terms and thus results in a high likelihood of Writer's Aid autonomously finding the necessary information. The system's performance of various search and formatting actions saves the writer time and effort identifying and creating bibliographic records and locating viewable versions of cited papers, enabling more efficient paper writing.

Besides being a natural framework for reasoning about goals and actions, planning offers advantages from the design and implementation standpoints. The declarative nature of planning-based interfaces allows extending the system by adding new types of user goals, new information sources, and new information retrieval actions *independently* of the existing code. As reported by Barish *et al.* [3] and confirmed by our own experience with Writer's Aid, once the planning structure is in place, designing, extending and modifying the system in response to users' requests required relatively little effort. This flexibility ensures that with more and more specialized searchable collections appearing on the Internet, Writer's Aid's repertoire of available search methods and sources will be easily augmented.

Initial laboratory user studies have shown Writer's Aid meets its design goals. In particular, most subjects (like many authors who are fluent in web technologies) ordinarily perform a sequence of online searches for bibliographic information and papers similar to those done by Writer's Aid. Even for such users, Writer's Aid's freeing them from doing these tasks and providing relevant information during the writing process in a timely manner was of significant help. An overwhelming majority of users found the system useful (some characterizing it as *very useful*), reflecting how often it was able to find papers the user intended to cite. Users found the interface intuitive and easy to learn. These results are all the more impressive because little attention was spent in fine-tuning the surface features of Writer's Aid; for example, the tested version of Writer's Aid did not use any advanced context-based rank-ordering of the search results. A further example of Writer's Aid's usefulness is the preparation of this paper: some of the references cited were identified using Writer's Aid and some of the bibliographic records and all inline citations were done by the system.

Writer's Aid is implemented on top of Carsten Dominik's Reftex package for the GNU Emacs editor, and the LaTeX and BibTeX document typesetting systems. The front end is implemented in Emacs Lisp, the planner in Allegro Common Lisp, and web access in WebL [9]. Writer's Aid is activated when the user opens a TeX document in the Emacs text editor.

After giving an example to illustrate the use and advantages of Writer's Aid, the paper enumerates characteristics of the bibliographic domain and task that underlie the design choices in Writer's Aid and then presents details of the system. The system description includes a discussion of the major issues that arise in building collaborative interfaces that utilize planning in domains with incomplete information, especially the implications for the system architecture and knowledge representation and planning methods. We briefly outline extensions to classical planning methods to meet the demands of collaborative interfaces in domains with properties like the Writer's Aid's. The paper then presents results of initial user studies, describes related work, and concludes with a discussion of possible future extensions to the system.

2. Overview and Example

To illustrate Writer's Aid's functions and main features, we will explore its use in the following scenario: An author, Ed is writing a paper on collaborative interfaces. He decides to refer to Kinny et al.'s article on teamwork but he does not recall the title of the paper nor where it appeared. He does not want to interrupt his writing to locate the paper, but he does want to scan the paper once it is found to make sure his claims about it are accurate.

Entering a citation command: Ed inserts a citation command with a special Emacs command. The system then prompts him to enter search parameters: keywords of the search and an indication of whether he wants only the bibliographic data on papers or the viewable versions as well. Ed enters *Kinny* and *team* as search keywords and selects the option of obtaining bibliographic records and viewable versions of relevant papers.

After a citation command is issued, a label resembling the LaTeX ordinary citation command is automatically generated and placed in the body of text. The label displays the type, keywords and status of the citation command as shown in Figure 1. The labels include the search keywords and type of search, a word indicating the status (*SEARCHING* or *DONE*) and the number of bibliographic records and viewable papers found in reference to the particular citation command; they may be updated to reflect the most recent findings by a simple user request.

While Ed continues writing (and inserting other citation commands) Writer's Aid plans and executes a search for the material he has requested. To make the search more efficient and better suited to Ed's needs, Writer's Aid limits the search for bibliographic information and papers to his preferred bibliographies and paper collections. Writer's Aid identifies preferred bibliographies semi-automatically at the time of installation by searching a user's home directory for his own bibtex files and inspecting his browser's bookmarks.

At installation time, Writer's Aid has identified as Ed's preferred bibliographies his own bibtex files and two on-line scientific collections: ResearchIndex and ACM Digital Library. It constructs a plan to query Ed's preferred bibliographic collections for the list of bibliographic records of papers that are related to the keywords *Kinny* and *team*. Once Writer's Aid has collected the list of relevant paper titles from Ed's bibtex file, ResearchIndex and ACM Digital Library it attempts to locate viewable version of each identified paper.

Writer's Aid's arsenal includes actions for parsing bibtex files; querying various digital repositories (currently NEC Research Institute's ResearchIndex and the ACM Digital Library) in search for papers, paper titles and authors' homepages; parsing homepages in search for papers with a given title; and downloading papers from a given URL.

Reviewing the results and selecting citation item: To view the data that Writer's Aid has collected in response to the citation command, Ed puts the cursor at the body of the citation command and issues a command to display the search results. The list of paper titles that has been compiled is displayed in a separate window, while the following options are a single keypress away: viewing and editing the bibtex record for an item; viewing the text of the paper, if it is available; selecting an item for citation. The prompt on the bottom of the selection buffer displays a help line with the commands for each option (see Figure 1).

Ed reviews the list, scanning some of the papers by issuing a view command until he identifies the paper he wants to cite, namely *"Planned Team Activity"*. He selects this paper with a single keystroke, and Writer's Aid ensures the citation is ready for compilation, that is, the appropriate bibliographic record is inserted in the bibliography file and the key for that record is placed in the text of the paper.

3. The Citation Application Domain

The Writer's Aid application has several characteristics that influenced the design of the system architecture and its constituent knowledge representation, reasoning, and planning systems. These requirements arise from two sets of characteristics: characteristics of the *interface*, that is, capabilities desired in the interaction with a person, and characteristics of the *domain*, that is the properties of references and citations. These characteristics also appear in many other applications for which collaborative interface systems would be beneficial, and hence their effect on system design are relevant beyond this particular application. We briefly describe these characteristics and their implications for the design and implementation of the collaborative interface system.

3.1 Interface Characteristics

We discuss three interface requirements in this section, along with their implications for the implemented system. These requirements were considered in the initial design of the collaborative interface and later refined given the observations and interviews from our pilot user studies.

Anytime editing/search/access capability: A key requirement of the interface is the seamless integration of the search and selection of papers for citation with the process of writing. A user can insert new citation commands and access possibly incomplete results of the search for any of the citation commands at any time while writing or editing a paper.

To guarantee the user fast and effective access to bibliographic information for all citations, information requests arising from citation commands are processed in a round-robin fashion, working on tasks in the order of increasing complexity. For instance, querying a bibliography for relevant bibliographic records is easier and faster than searching for the viewable version of a paper. As a result, Writer's Aid first attempts to locate the bibliographic records for all citations, and postpones attempting to satisfy goals related to obtaining their viewable versions.[1]

Availability of partial results and search status: A user can access the results of a search and make a selection at any time, even when the search has not yet completed. When using Writer's Aid, a person's primary task, and hence focus, is typically on writing the paper. As a result, users usually do not explicitly monitor the progress of the system. However, Writer's Aid informs the user of the progress of the search by updating the body of the citation command appearing in the text of the paper (see Figure 1). The display of search-status information is helpful in two ways: It enables early detection of queries that produce no matches (allowing reformulation of the citation command), and it is a way to inform users about completion status of a citation, before they start reviewing and selecting from the list of papers.

3.2 Domain Characteristics

The domain of Writer's Aid has two characteristics that directly affect the types of technology used in the underlying system, both relating to the *incompleteness* of the information possessed by the system.

A major challenge to systems design is the *inherent incompleteness* of information about Writer's Aid's domain: bibliographic records, papers, their locations, keywords. A complete description of this domain cannot be provided a priori and can never be fully acquired. Rather, the system must be able to represent partial information and to reason about acquiring missing information that is necessary to satisfy the planning goals related to a user's citation needs.

Further, Writer's Aid's domain knowledge has *local incompleteness*; it is incomplete even with respect to properties of the objects the system knows about. For instance, it may not know which papers have a particular keyword in their abstracts or where viewable versions of a paper are located. As a result, actions in the bibliographic domain rely heavily on information gathering to in turn affect the actions to be

[1]However, a user can override this default and can focus Writer's Aid specifically on getting a particular paper by using a special *immediate* citation command. The search for materials related to immediate citation is not abandoned until all possibilities are attempted, that is, until all related planning goals are either satisfied or found unsatisfiable.

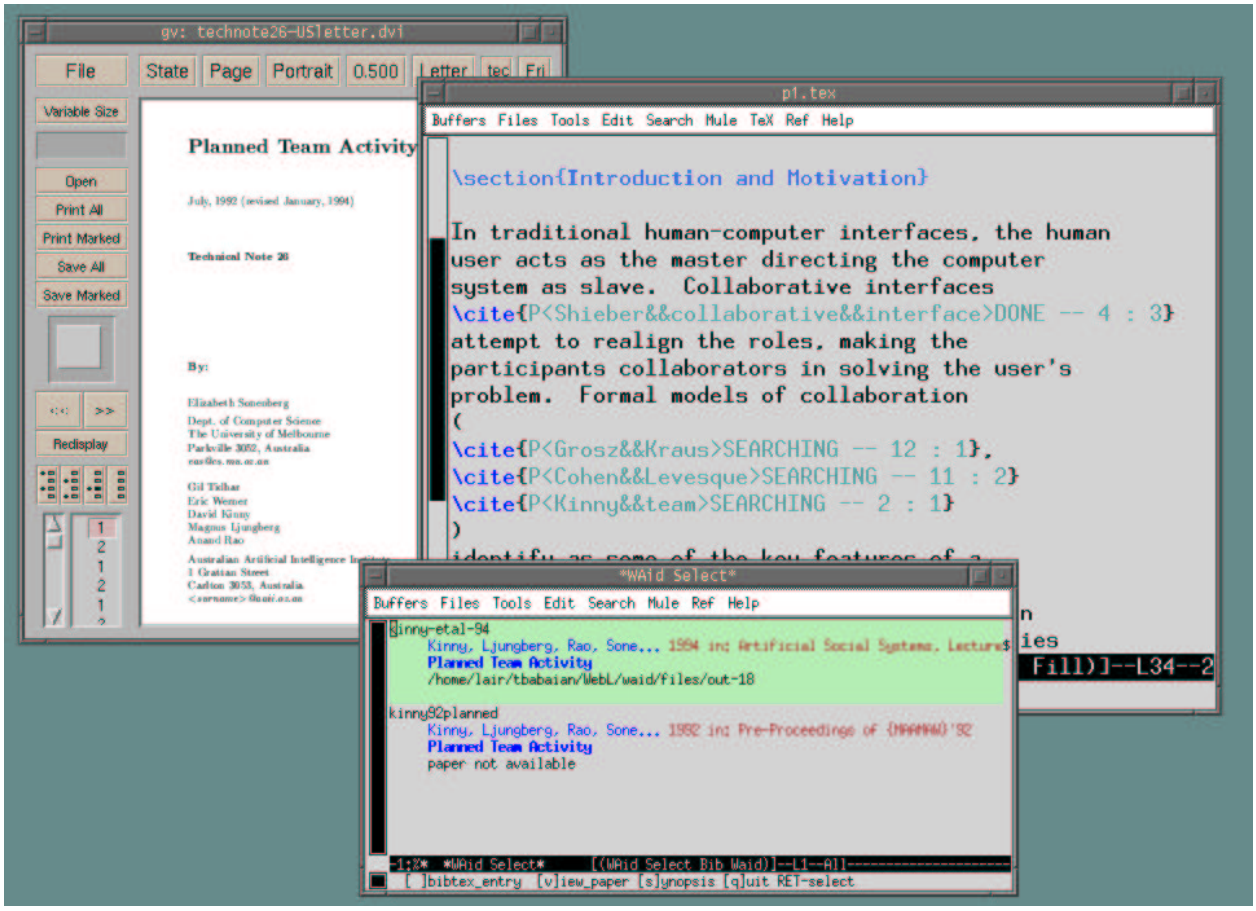

Figure 1: A snapshot of Writer's Aid. In the middle Emacs window, the user has entered a set of citations in the text of a paper. The body of the citation command displays the status of the searches, the first of which is completed. The user is browsing the paper list from one of the incomplete searches in the front window. The rear window is showing the first paper from the list, retrieved by a single keystroke.

taken subsequently. For example, the results of a query for relevant papers may determine which viewable versions of papers the system acquires. The system must therefore be able to interleave information acquisition and planning; this is a special case of interleaved planning and plan execution.

Classical planning techniques are insufficient to handle these properties of the domain. To address inherent incompleteness, Writer's Aid uses an expressive yet tractable logic, PSIPLAN[2], which allows efficient representation of incomplete information. To address local incompleteness and allow for information gathering, Writer's Aid deploys a novel method for combining planning with execution of incomplete plans, which we call *planning with hypotheticals*. These important technical aspects of our solution are described in a later section.

The domain characteristics interact with the interface characteristics. For instance, since Writer's Aid begins with little knowledge about papers relevant to the user's request, a substantial amount of information gathering may be required to satisfy a user's requests. Because most of the information is obtained from remote sources over the Internet, it may take considerable time to identify, locate and download all of this information. On the other hand, it is very likely that the user will be satisfied with only partial results of the search, as conventional search engines often provide only partial results. To make partial results quickly available to the user (an important interface characteristic), Writer's Aid's design includes (i) formulation of the information request into a set of goals, processed in order of increasing likelihood of relevancy to the user, (ii) initial goal reduction to account for already available information, and (iii) round-robin processing of information requests in order of increasing search complexity. These features are described in more detail in the next sections of the paper.

4. Architecture Overview

The architecture of Writer's Aid contains the following three major components in addition to a front-end Emacs interface:

- **State of Knowledge** (**SOK**) and **Goal** (**G**) **databases**: The **SOK** database contains Writer's Aid's knowledge about the user's preferences and the world of bibliographies, papers and paper sources. The **G** database records the system's goals.

- The **Reasoning module** (**R**): This module handles goal reduction with respect to the **SOK** database.

- The **Planning Problem Manager** (**PPM**): This module constructs and manages planning problems arising from a user's citation requests. It includes a planning and execution module, PSIPOP-SE (PSIplan-based Partial Order Planner with Sensing and Execution), which constructs and executes individual plans.

In brief, Writer's Aid uses these components to handle a user's citation command as follows: The command itself results in a goal being posted to the goal database **G** and the goal reduction module **R** being invoked as a separate thread. **R** consults the **SOK** database and computes the part of the goal that is already accomplished and the part that still remains to be achieved. It places the latter onto **G**, passing it to the planning problem manager, **PPM**. The **PPM** module creates an instance of a planning problem and hands it to the planner, PSIPOP-SE, which either constructs and executes a plan or reports failure if the planning problem is unsolvable.

Upon executing the plan actions, Writer's Aid updates the **SOK** database to reflect all changes in knowledge. For example, additional knowledge generated by an information-gathering action is added. Upon completion of its part, **PPM** removes the goals that were satisfied from the goal agenda, records the failure for the (sub)goals that **PPM** failed to achieve, and proceeds with the next goal.

When a user issues a command to view a list of records and papers corresponding to a citation command, this information is derived from the **SOK**, formatted, and presented in a separate window for browsing.

4.1 SOK and Goal Formulation

All of Writer's Aid's knowledge about the world is contained in the **SOK** database. As discussed above, this knowledge is assumed to be correct but incomplete. Since the system cannot have access to a complete description of the world, it must be able to effectively represent, reason, and plan with incomplete knowledge.

Writer's Aid uses the PSIPLAN language [2] which enables efficient representation of an agent's incomplete knowledge about the world and knowledge goals and has an associated knowledge update procedure that is efficient. As described in the language specification [2], PSIPLAN entailment is sound, complete, and takes only polynomial time in the size of the agent's **SOK** database. Alternative planning representations are either intractable in the general case, or, as with the tractable LCW (locally closed world) representation [6], lack completeness and sometimes discard correct information. Precision in reasoning about the world in the presence of the unknown bears directly on the ability to have non-redundancy of information gathering; it is thus especially critical for a system that uses costly (time-consuming) information-gathering actions. Incompleteness of reasoning may cause failure to construct all possible plans, which is also problematic for a collaborative agent.

PSIPLAN formulas are either ground atoms over function-free terms, universally quantified negated clauses with exceptions, or knowledge propositions. For example the statement

> *The only bibliographies preferred by Ed are the digital library of the ACM, and maybe the ResearchIndex database.*

is represented in PSIPLAN by the following two propositions:[2]

1. *ACM's digital library is a preferred bibliography*, which is represented by a ground atom:

$$PrefBib(ACM)$$

2. *Nothing is a preferred bibliography except for ACM and the ResearchIndex*, which is expressed as the following quantified negated clause with exceptions:

$$\forall b \neg PrefBib(b) \vee b = ACM \vee b = RI$$

To represent that a value of a certain proposition is known, PSIPLAN uses knowledge propositions; $KW(PrefBib(ACM))$ denotes that the agent knows the truth value of $PrefBib(ACM)$, that is, the agent knows whether ACM is a preferred bibliography.

To represent the user's goals, Writer's Aid extends PSIPLAN to handle *implication goals* of the form $\forall \vec{x}\, \exists \vec{y}\, P(\vec{x}, \vec{y}) \implies Q(\vec{x}, \vec{y})$, where $\vec{x}$ and $\vec{y}$ are sets of variables, and both P and Q are conjunctions of atoms.

A user's request to obtain papers relevant to subject Y is formulated as the following goal:

> *For each paper that is relevant to subject Y according to some bibliography preferred by Ed, get that paper and get the bibliographic record for it.*

This goal is instantiated as three separate PSIPLAN goal formulas. The first goal is to obtain all papers and bibliographic records of papers containing keywords Y in the title and referenced in the user's own local bibliographic collections:

$$\begin{array}{l}\forall p\, \exists b\, PrefBib(b) \wedge LocalBib(b) \wedge InCollection(p,b) \wedge \\ TitleUses(p,Y) \implies Got(p) \wedge GotBib(p)\end{array} \quad (1)$$

The second goal extends the first to *all* of the user's preferred bibliographic collections.

$$\begin{array}{l}\forall p\, \exists b\, PrefBib(b) \wedge InCollection(p,b) \wedge \\ TitleUses(p,Y) \implies Got(p) \wedge GotBib(p)\end{array} \quad (2)$$

The last goal is to obtain all papers containing keywords Y in the text, rather than in the title.

$$\begin{array}{l}\forall p\, \exists b\, PrefBib(b) \wedge InCollection(p,b) \wedge \\ TextUses(p,Y) \implies Got(p) \wedge GotBib(p)\end{array} \quad (3)$$

The first goal is entailed by the second, which is entailed by the third; thus, the set of papers required by the first goal is subsumed by the set of second goal's papers, which, in turn, is subsumed by the third goal (since a title is a part of the text). However, these three goals are posted and processed in the order presented above to explicitly prioritize

[2]In this section, we use the following predicates: $PrefBib(b)$ denotes that b is a preferred bibliography; $LocalBib(b)$ denotes that b is a locally stored bibtex bibliography; $InCollection(p,b)$ denotes paper p being in collection of bibliography b; $TitleUses(p,Y)$ denotes that keywords Y occur in p's title (where by title we mean a combination of the title and author names); $TextUses(p,Y)$ denotes that keywords Y occur in p's full text including the title and author fields; $Got(p)$ and $GotBib(p)$ denote, respectively, that paper p and its bibliographic record are stored locally.

the search for papers that are more likely to be in the desired set. Writer's Aid is able to accomplish this incremental processing without doing redundant searches for the same information by saving in the SOK the information acquired during its attempts to satisfy the first and second goals.

4.2 Goal Reduction

Once a goal is posted to the goal database **G**, the *goal reduction module* **R** handles the processing of the goal. **R** chooses a goal from **G**, reducing it with respect to the **SOK**, and passing it to **PPM**. When the planner returns, **R** records success or failure in achieving the goal, and proceeds to the next one.

For simplicity of presentation, we abbreviate a conjunction of predicates occurring in the left hand side of goals (1-3) above by a metapredicate $Rel(p, b, Y)$ to indicate that a paper p is *relevant* to keywords Y according to bibliography b, and drop $GotBib(p)$ from the right hand side. Thus, the goal with which we are concerned is

$$g = \forall p \, \exists b \, PrefBib(b) \wedge Rel(p, b, Y) \implies Got(p) \qquad (4)$$

To satisfy this goal, it is first necessary to find all papers that are relevant to Y according to some preferred bibliography and then, for those papers only, construct a plan of obtaining them. Thus, **R** transforms g into two goals in PSIPLAN's base language:

1. finding out the truth value of the conjunction $PrefBib(b) \wedge Rel(p, b, Y)$ for all possible values of b and p, i.e.

$$g_1 = \forall p \, \forall b \, KW(PrefBib(b) \wedge Rel(p, b, Y)),$$

   and, after g_1 is achieved,

2. instances of $Got(p)$ corresponding to all values of p for which $PrefBib(b) \wedge Rel(p, b, Y)$ is true.

R places g_1 as the next goal of **G** and further reduces it with respect to **SOK** to identify the part that is not already known (e.g., as a result of previously executed information-gathering actions). This computation corresponds to a special PSIPLAN operation, called *extended difference*, denoted $\dot{-}$. Given PSIPLAN propositions A and B, $A \dot{-} B$ is the set of propositions of A that are not entailed by B. **R** reduces any goal g by computing the extended difference $g \dot{-} \mathbf{SOK}$. For example, given an information goal g_1 and an **SOK** that contains information that nothing is a preferred bibliography except for possibly the ACM digital library and the ResearchIndex, **R** deduces that the only remaining information goals are

$$g_2 = \forall p \, KW(PrefBib(ACM) \wedge Rel(p, ACM, Y)),$$
$$g_3 = \forall p \, KW(PrefBib(RI) \wedge Rel(p, RI, Y)).$$

passing g_2 and g_3 to the **PPM**.

Such reduction of g, if not done prior to planning, would need to be carried out while planning to achieve this goal inside the planner itself. However, in our formalism no information ever gets lost, so that such early separation of yet unknown facts from those already known is an advantage, because it identifies *exactly* what goal the planner is working to achieve, and the user can access that information while the planner is working on the goal. The advantage becomes even more apparent if we consider having multiple agents working to achieve the goal. In such cases, reducing the goal initially prevents redundant computation.

4.3 Managing Planning Problems

Once the reduced goal is computed, it is passed to **PPM**, the Planning Problem Manager, which takes care of creating, prioritizing, solving, and keeping track of the status of multiple planning tasks arising from goals adopted by Writer's Aid. **PPM** consists of two major components: a list of *planning problems*, and a *planning algorithm* PSIPOP-SE, which constructs solution plans for individual planning problems.

When a goal is passed to **PPM**, a new planning problem is created and passed to PSIPOP-SE, which searches for a solution plan, and returns the result. Each planning problem is a structure that records a planning goal, its solution, and the overall status of the planning problem, which is one of *open, done, unsatisfiable*. *Open* problems are those for which the solution plan has not been found, yet the goal has not yet been found to be unsatisfiable. If a solution plan is found and successfully executed, **PPM** removes the planning problem from the list of open problems and places it on the *done* list. If a solution is found but an action execution failure occurs, the failed action instance is recorded and never used again by the planner; the planning problem remains on the open list until the planner establishes that no alternative course of action exists. *Unsatisfiable* problems are those that have unachievable goals.

Iterative Deepening in Hypotheticals: To guarantee step-by-step processing, and availability of partial results of the search for all of the user's requests as motivated earlier, **PPM** processes open problems in a round-robin fashion, gradually increasing the maximum complexity level of finding and executing the solution plan. To implement the gradual increase of solution complexity, **PPM** performs *iterative deepening in hypotheticals*. A *hypothetical* is a partial plan that hypothesizes on the value of an unknown proposition or subgoal. For example, having no information on the location of a paper, the planner may adopt a hypothesis that the paper is available from a certain collection, and verify the information by querying the collection. An example of a plan with two hypotheses is a plan that hypothesizes that a paper is available from the author's homepage, and then, having no information about the author's homepage, hypothesizes that the URL for the homepage can be found from a known index.

By verifying a hypothesis via execution of a sensing action, the planner eventually collects enough information, and thus reduces the incompleteness of the knowledge enough to find a solution plan or find the goal unsatisfiable.

PPM maintains a list of all open problems, processed in a loop. At each cycle of the loop **PPM** attempts to find a solution for each open problem in turn, increasing the maximum allowed number of hypotheses in a solution plan when necessary, and executes the plan until the processing is completed and the problem is removed from the open list.

This combination of iterative deepening in hypotheticals with round-robin processing of planning problems enables effective time sharing between the user's goals, which is necessary for providing partial results on many user requests si-

multaneously, and avoiding the bottlenecks of searching for a hard to find paper, which may not be the one desired by the user.

5. Evaluation

We performed a pilot study with two users, followed by a user study involving eleven subjects. Most of the subjects were Harvard University students and postdocs; eleven are computer scientists, one a physicist. Most, though not all, of the subjects were familiar with Emacs and had previously written papers using LaTeX and BibTeX.

The subjects were shown a brief, two-minute demonstration of the system; they were then given a printed tutorial[3] and asked to follow the steps of the tutorial. The subjects were next asked to write a paragraph or two of text in the area of their expertise involving citations, using Writer's Aid. All the subjects used the same local bibliography collection, which overlapped with some of the citations some subjects desired to make, but most of the bibliographic records required by the authors were dynamically collected from ResearchIndex.

To our surprise, even without access to the writer's personal BibTeX database, but using only ResearchIndex as another preferred bibliography and the (dynamically located) authors' homepages in the search for papers, Writer's Aid was able in most cases to successfully locate at least bibliographic records for the papers. The success rate for finding viewable versions was more modest, but users still found the system very helpful. We expect a higher number of papers could be found by expanding the set of sources to include more online collections.

After the test, subjects completed a questionnaire allowing freeform answers to the following questions:

1. How hard was it to learn to use the Writer's Aid?
2. Was it useful? Would you use it for writing papers?
3. Which modifications to the functionality/interface of Writer's Aid would you recommend?

Some users were later interviewed to clarify their responses to Question 3.

The success of Writer's Aid is indicated by the answers to the Question 2. To the first part *"Was* Writer's Aid *useful?"* the replies were: *very useful* (3), *useful* (7), *moderately useful* (1). To the question *"Would you use it for writing papers?"* ten users answered *yes.* (The single dissenting user explained that he would not trust any online source with a bibliographic record, so he would manually verify all such records anyway, making Writer's Aid redundant in his mind.)

To the question *How hard was it to learn to use* Writer's Aid*?* 4 users answered *very easy*, 2 *easy*, and 5 *reasonably easy* or *not hard*.

In response to Question 3, users suggested adding morphology-aware search, automatic spell checking of keywords, an ability to add a record to the personal bibliographic collection without citing it, and minor alterations to the window interface. We are planning to implement some of these features in the next version of Writer's Aid.

[3]The tutorial is available at `http://www.eecs.harvard.edu/~tbabaian/waid/tutor.ps`.

6. Related Work and Future Directions

Research presented in this paper has connections to work in several areas, most notably AI-based collaborative interfaces, information integration systems and Internet search.

Like many other information integration systems, Writer's Aid takes advantage of the breadth of bibliographic information available on the web. BIG [10] integrates several AI technologies, including resource-bounded planning and scheduling to conduct an offline search for information on software packages based on a client's specification. Barish *et al.* [3] report on a query-planning-based system, called TheaterLoc, that searches online movie-related databases in real time in response to users' queries. Writer's Aid differs from these and other planning-based information-retrieval systems [11] in carrying out its activities in the context of collaboration with a user in the ongoing writing process, so that this writing process provides context for interpreting the information request. Writer's Aid is also distinguished from other planning-based information retrieval systems by the capabilities it incorporates for interleaved planning and execution, crucial for integrating information-gathering into the planning process.

Collagen [15] is a middleware package based on a theory of collaboration in dialogue [12]; it provides a means for creating interfaces that participate in dialogues with users about their goals and beliefs, suggesting possible courses of action based on the available library of act recipes. Collagen does not include capabilities for automated reasoning about goal achievement beyond the use of a fixed set of recipes. Thus, it lacks Writer's Aid's ability to satisfy user goals from almost any initial state using a variety of dynamically created courses of actions. Collagen's collaborative strength is its ability to work with the user through a process, known (via a recipe library) to the system, leading to achievement of the user's goal. The focus in Writer's Aid is on another system capability important for collaboration, namely, the ability to plan for and carry out autonomously a complex task that otherwise would have to be done by the human, and integrating the activities of the system-partner with those of the user in a non-intrusive and efficient manner.

Other work has explored the use of context in information retrieval. Watson [4] is intended to work with its user proactively downloading and suggesting information it regards as relevant to a document that the user is currently editing or viewing. Watson creates a search query based on the text and the structure of the document, but not related to any specific user request. However, the user study of Watson [4] evaluated the utility of information provided by Watson statically; it did not involve the system working "alongside" a user. As a result, the appropriateness of Watson's search results in interactive use was not evaluated in that study. In contrast, Writer's Aid takes seriously the fact that when users delegate to a system the task of finding information needed to complete a task (or satisfy a user's goal), the usefulness of the system depends critically on the relevancy of the information retrieved by the system and on the results being available in a timely manner. Otherwise, the time it takes the user to sift through irrelevant information or the time spent waiting for the results may outweigh the time the user saves by not performing the search himself. These performance characteristics in Writer's Aid are ensured by the system adopting the precisely specified *user's* search goal and using information sources that are directly related to

a well defined set of data items such as papers and bibliographic records.

In the future, we plan to extend Writer's Aid to incorporate the context of a citation request for more efficient search and ranking of the results. Another direction we have started to explore is adding the user as a source of information about his or her own preferences and knowledge of relevance of various online collections to the subject of a paper. Such personalization tasks can be stated declaratively via a set of knowledge goals and satisfied by an action of querying the writer, when this information becomes necessary. This representation separates personalization of the interface from its overall architecture, making it more easily adjustable. It also leads to preference elicitation that occurs within the context of a particular task.

7. Conclusion

We have presented a writer's assistant system that works collaboratively with a user, achieving the necessary flexibility of behavior through explicit representation, reasoning, and planning with respect to goals and domain knowledge. Collaborativeness is embodied in the system's commitment to shared goals of producing accurate, well-formed citations; a division of labor in which each participant contributes according to natural capabilities, pursuing all known avenues to accomplish those goals; and communication between the parties in both directions, the user providing query information and bibliographic choices to the system, the system providing query status and gathered information to the user.

The use of planning technology to implement collaborative interfaces places new requirements on the knowledge representation and planning methods. We presented a set of extensions to classical planning representations and techniques to satisfy these requirements. In particular, the use of an expressive, yet precise and tractable formalism for knowledge representation, PSIPLAN, and the addition of hypothetical planning to integrate domain actions with sensing actions and interleaved execution, were crucial to the implementation of the collaboration.

We conducted a laboratory user study to examine the effectiveness of the system. The results indicate the success of this particular interface and its implementation. Users characterized it as a useful and easy-to-learn tool that they would like to have for academic writing.

8. Acknowledgements

The research reported in this paper was supported by National Science Foundation grants IRI-9618848 and IIS-9978343 to Harvard University. The authors thank Luke Hunsberger, Wheeler Ruml and Christian Lindig for their assistance in developing the system and for helpful comments on the paper, and all participants of the user study.

9. References

[1] Yigal Arens, Craig A. Knoblock, and Wei-Min Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems - Special Issue on Intelligent Information Integration*, 6(2/3):99–130, 1996.

[2] Tamara Babaian. *Knowledge Representation and Open World Planning Using ψ-forms*. PhD thesis, Tufts University, 2000.

[3] Greg Barish, Craig A. Knoblock, Yi-Shin Chen, Steven Minton, Andrew Philpot, and Cyrus Shahabi. The TheaterLoc virtual application. In *AAAI/IAAI*, pages 980–987, 2000.

[4] Jay Budzik and Kristian Hammond. User interactions with everyday applications as context for just-in-time information access. In *Proceedings of IUI2000*, 2000.

[5] P. Cohen and H. Levesque. Teamwork. *Nôus*, 25:487–512, 1991.

[6] O. Etzioni, K. Golden, and D. Weld. Sound and efficient closed-world reasoning for planning. *Artificial Intelligence*, 89(1–2):113–148, January 1997.

[7] Barbara J. Grosz and Sarit Kraus. The Evolution of Shared Plans. In A. Rao and M. Wooldridge, editors, *Foundations of Rational Agency*, pages 227–262. Kluwer Academic Press, 1999.

[8] D. Kinny, M. Ljungberg, A. S. Rao, E. Sonenberg, G. Tidhar, and E. Werner. Planned team activity. In C. Castelfranchi and E. Werner, editors, *Artificial Social Systems, Lecture Notes in Artificial Intelligence (LNAI-830)*, Amsterdam, The Netherlands, 1994. Springer Verlag.

[9] Thomas Kistler and Hannes Marais. WebL – a programming language for the web. *Computer Networks and ISDN Systems*, 30(1–7):259–270, 1998.

[10] V. Lesser, B. Horling, F. Klassner, A. Raja, T. Wagner, and S. Zhang. Big: An agent for resource-bounded information gathering and decision making. *Artificial Intelligence*, 118:197–244, 2000.

[11] A. Levy and D. Weld, editors. *Artificial Intelligence*, volume 118. Elsevier Science, 2000.

[12] K. E. Lochbaum. A collaborative planning model of intentional structure. *Computational Linguistics*, 24, 1994.

[13] C. Ortiz and B.J. Grosz. Interpreting information requests in context: a collaborative web interface for distance learning. Autonomous Agents and Multi-Agent Systems Journal, to appear, 2002.

[14] C. Rich and C. Sidner. Segmented interaction history in a collaborative interface agent. In *Proceedings of IUI97*, 1997.

[15] C. Rich, C. Sidner, and N. Lesh. Collagen: Applying collaborative discourse theory to human-computer interaction. *AI Magazine, Special Issue on Intelligent User Interfaces*, 2001.

[16] K. Ryall, J. Marks, and S. Shieber. An interactive constraint-based system for drawing graphs. In *Proceedings of UIST*, 1997.

[17] S. Shieber. A call for collaborative interfaces. *ACM Computing Surveys*, 28A (electronic), 1996.